Practical machine learning report

Data Science specialization Coursera

Report by Carlos Neves

Introduction

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify *how much of a particular activity they do*, but they rarely quantify *how well they do it*.

Aim

In this project, 6 participants were asked to perform barbell lifts correctly and incorrectly in 5 different ways. The data from **accelerometers** on the **belt, forearm, arm, and dumbell** will be used to predict *how well* the barbell lifts were performed. The data was used for this analysis was this training set and this test set. These data are based on the study Human Activity Recognition

Analysis

Initial setting

- To guarantee reproducibility, the seed 452 was set.
- The analysis requires the package caret (Classification And Regression Training) for model building
- The data from the study was loaded and the training data was divided into train and cross-validation datasets.

```
# for reproducibility
set.seed(452)

# required package
suppressMessages(library(caret))

# load the raw data
train.data<-read.csv("pml-training.csv",header=TRUE)
test.data<-read.csv("pml-testing.csv",header=TRUE)

# divide the training data for cross-validation
intrain<-createDataPartition(train.data$classe,p=0.7,list=FALSE)
train<-train.data[intrain,]
cross<-train.data[-intrain,]
```

Data processing

Since I am interested in predicting the quality of the activity by using the data from the *accelerometers* on the belt, forearm, arm, and dumbell, the train data set was simplified by taking only the variables corresponding to the accelerometer values.

```
# select only the accelerometer data
accel<-grep("accel",names(train))
train<-train[,c(accel,ncol(train))]
head(train)
##    total_accel_belt var_total_accel_belt accel_belt_x accel_belt_y
## 1                 3                   NA          -21            4
## 2                 3                   NA          -22            4
## 4                 3                   NA          -22            3
## 7                 3                   NA          -22            3
## 9                 3                   NA          -20            2
## 10                3                   NA          -21            4
##    accel_belt_z total_accel_arm var_accel_arm accel_arm_x accel_arm_y
## 1            22              34            NA        -288         109
## 2            22              34            NA        -290         110
## 4            21              34            NA        -289         111
## 7            21              34            NA        -289         111
## 9            24              34            NA        -288         109
## 10           22              34            NA        -288         110
##    accel_arm_z total_accel_dumbbell var_accel_dumbbell accel_dumbbell_x
## 1         -123                   37                NA             -234
## 2         -125                   37                NA             -233
## 4         -123                   37                NA             -232
## 7         -125                   37                NA             -232
## 9         -122                   37                NA             -232
## 10        -124                   37                NA             -235
##    accel_dumbbell_y accel_dumbbell_z total_accel_forearm var_accel_forearm
## 1                47             -271                  36                NA
## 2                47             -269                  36                NA
## 4                48             -269                  36                NA
## 7                47             -270                  36                NA
## 9                47             -269                  36                NA
## 10               48             -270                  36                NA
##    accel_forearm_x accel_forearm_y accel_forearm_z classe
## 1              192             203            -215      A
## 2              192             203            -216      A
## 4              189             206            -214      A
## 7              195             205            -215      A
## 9              193             204            -214      A
## 10             190             205            -215      A
```

As some of these variables seem to have a great number of missing values (NA), incomplete variables were also excluded

```
# find incomplete variables
temp<-numeric()
for (i in 1:ncol(train)) {
  if(sum(is.na(train[,i]))!=0) {
    print(paste(i,names(train)[i],sep="   "))
    temp<-append(temp,i)
      }
}
## [1] "2   var_total_accel_belt"
## [1] "7   var_accel_arm"
## [1] "12   var_accel_dumbbell"
```

```
## [1] "17   var_accel_forearm"
```

The incomplete variables correspond to the variance of the values measurements. As the mean values have more predictablility value than the variance of these values, the incomplete variables were removed.

```
train<-train[,-temp]
names(train)
##  [1] "total_accel_belt"     "accel_belt_x"         "accel_belt_y"
##  [4] "accel_belt_z"         "total_accel_arm"      "accel_arm_x"
##  [7] "accel_arm_y"          "accel_arm_z"          "total_accel_dumbbell"
## [10] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [13] "total_accel_forearm"  "accel_forearm_x"      "accel_forearm_y"
## [16] "accel_forearm_z"      "classe"
```

Exploratory data analysis

The remaining variables were ploted to check their influence in the exercise quality (*classe*). Exercise quality is described as Class A if it corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes observed.

```
# explore the influence of each variable
featurePlot(x=train[,1:4],y=train$classe,plot="box")
```

```
featurePlot(x=train[,5:8],y=train$classe,plot="box")
```

```
featurePlot(x=train[,9:12],y=train$classe,plot="box")
```

```
featurePlot(x=train[,13:16],y=train$classe,plot="box")
```

All of the total measurements (*total_accel*) from the accelerometers seem to be unrelated to the *classe* of exercise

```
# remove "total" measurements
train<-train[,c(-1,-5,-9,-13)]
names(train)
##  [1] "accel_belt_x"     "accel_belt_y"     "accel_belt_z"
##  [4] "accel_arm_x"      "accel_arm_y"      "accel_arm_z"
##  [7] "accel_dumbbell_x" "accel_dumbbell_y" "accel_dumbbell_z"
## [10] "accel_forearm_x"  "accel_forearm_y"  "accel_forearm_z"
## [13] "classe"
```

Model building

The chosen variables were converted into numeric values.

```
# set all variables as numeric
for (i in 1:(ncol(train)-1)) {
    train[,i]<-as.numeric(train[,i])
}
```

Having simplified the train data and selected the most relevant variables, a random forest model (with k-fold cross-validation) was chosen to predict how well the the barbell lifts were performed, because a linear relationship is not apparent for these categorical values.

```
model<-train(classe~.,data=train,method="rf",prox=TRUE,trControl = trainControl(method = "cv",number=5))

model
```

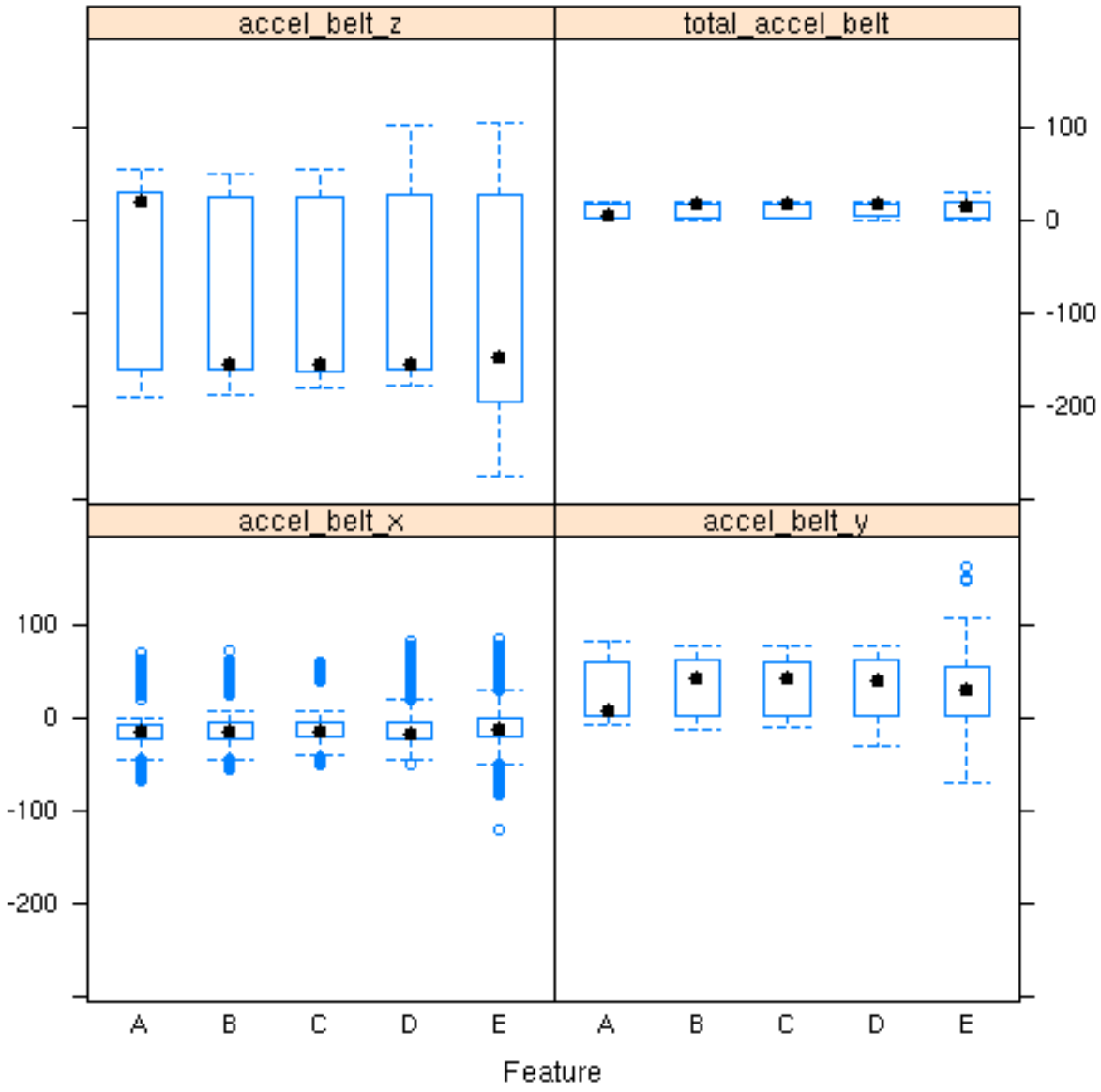

Figure 1: plot of chunk unnamed-chunk-5

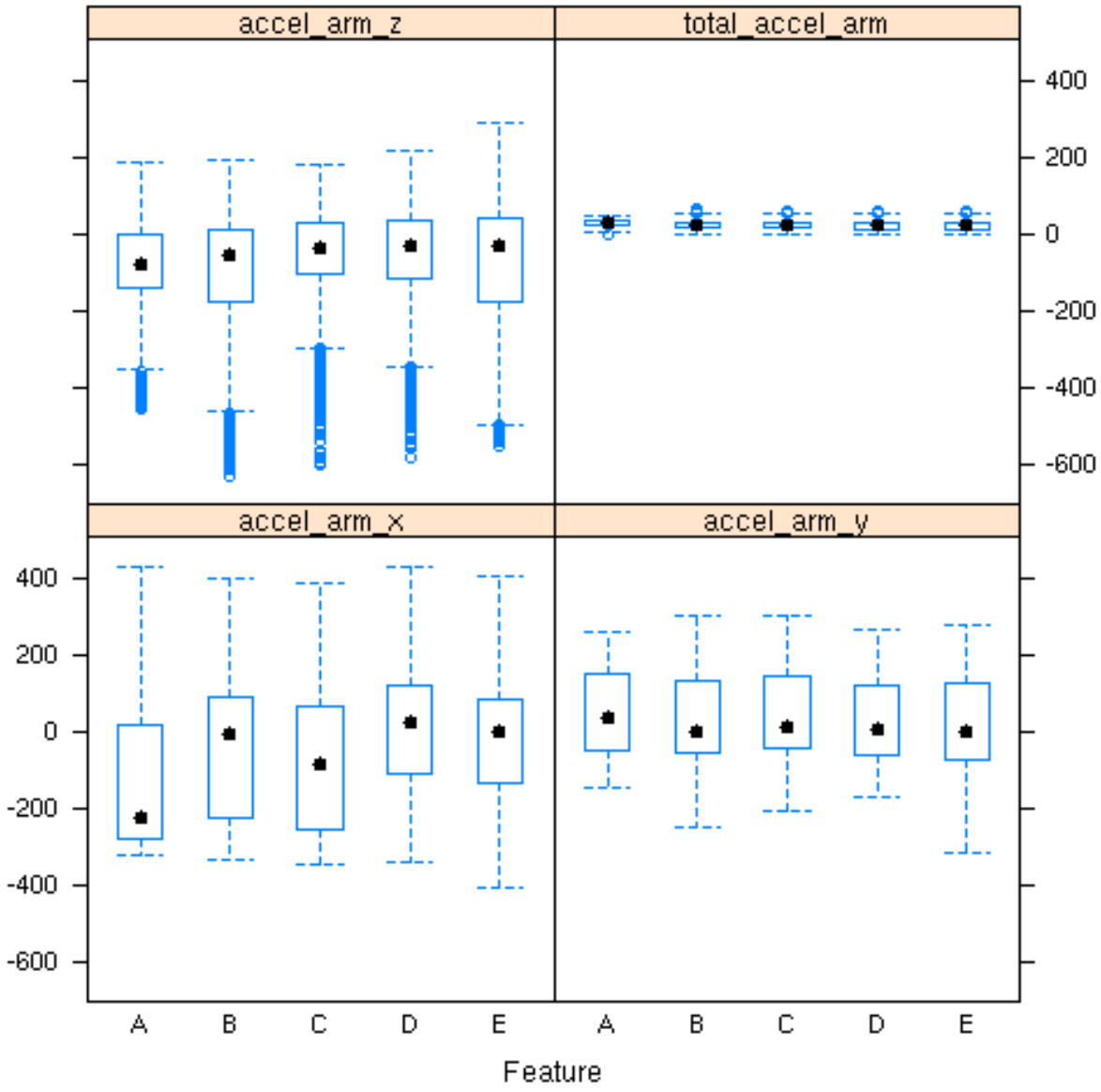

Figure 2: plot of chunk unnamed-chunk-5

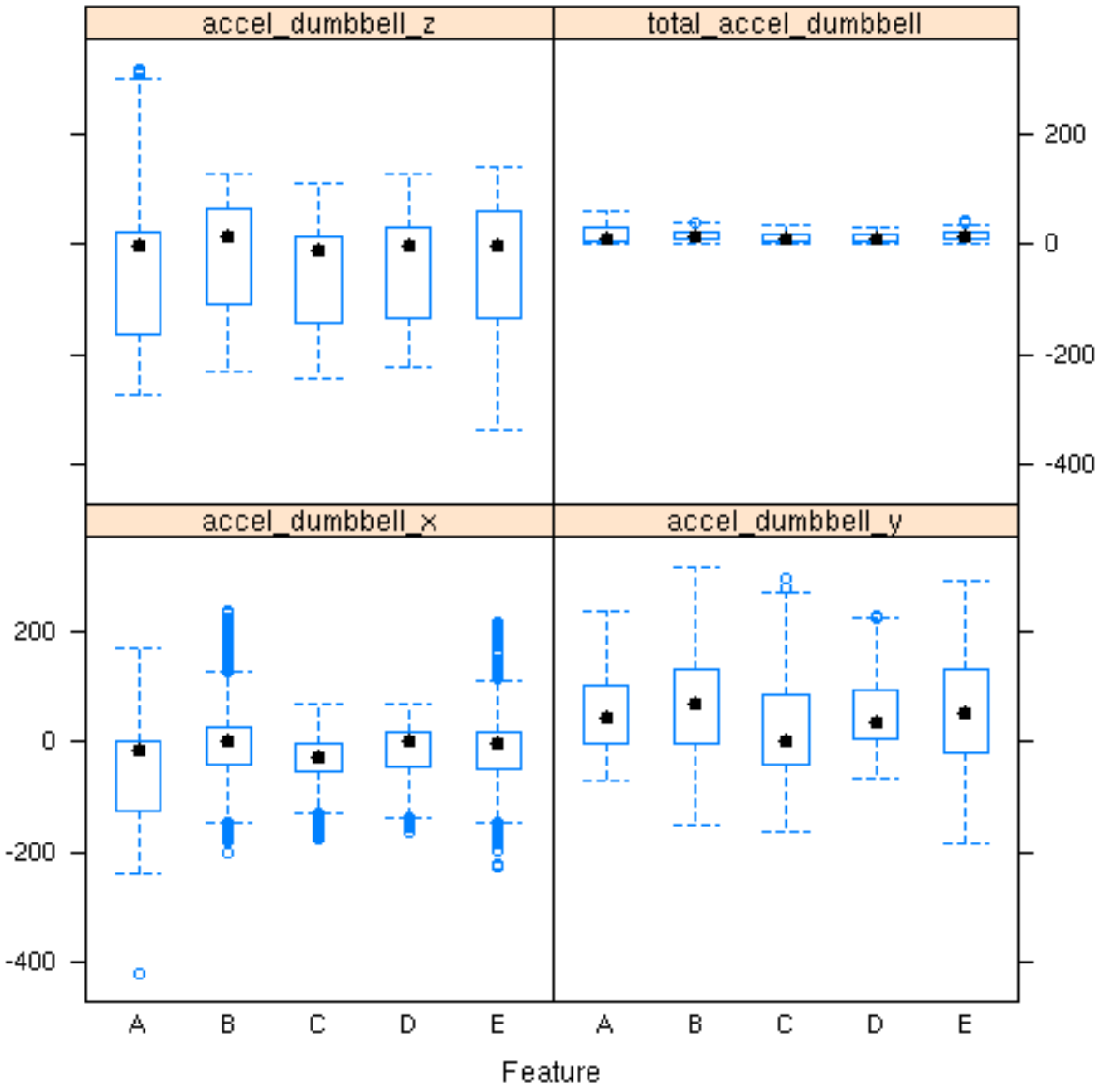

Figure 3: plot of chunk unnamed-chunk-5

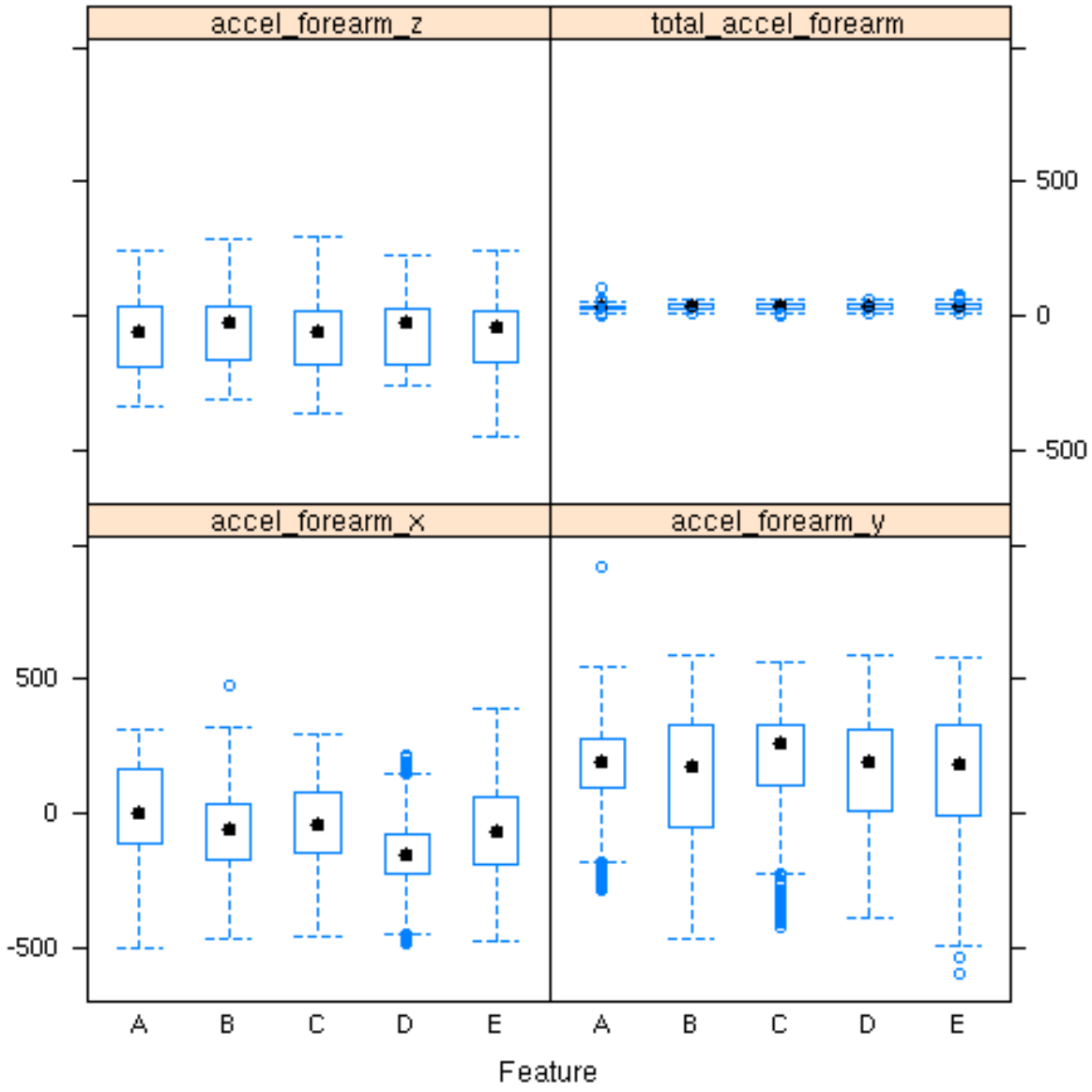

Figure 4: plot of chunk unnamed-chunk-5

```
## Random Forest
##
## 13737 samples
##    12 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10989, 10990, 10990, 10989, 10990
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9389240  0.9227291
##    7    0.9341192  0.9166679
##   12    0.9186142  0.8970265
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

Cross valiation/Out of sample error

The *in-sample-error* was measured by comparing the prediction of the model with the real value of the of the train data

```
table(train$classe,predict(model,train))
##
##        A    B    C    D    E
##   A 3906    0    0    0    0
##   B    0 2658    0    0    0
##   C    0    0 2396    0    0
##   D    0    0    0 2252    0
##   E    0    0    0    0 2525
confusionMatrix(train$classe,predict(model,train))$overall
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      1.0000000      1.0000000      0.9997315      1.0000000      0.2843416
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
confusionMatrix(train$classe,predict(model,train))$byClass
##          Sensitivity Specificity Pos Pred Value Neg Pred Value Precision
## Class: A           1           1              1              1         1
## Class: B           1           1              1              1         1
## Class: C           1           1              1              1         1
## Class: D           1           1              1              1         1
## Class: E           1           1              1              1         1
##          Recall F1 Prevalence Detection Rate Detection Prevalence
## Class: A      1  1  0.2843416      0.2843416            0.2843416
## Class: B      1  1  0.1934920      0.1934920            0.1934920
## Class: C      1  1  0.1744195      0.1744195            0.1744195
## Class: D      1  1  0.1639368      0.1639368            0.1639368
## Class: E      1  1  0.1838101      0.1838101            0.1838101
##          Balanced Accuracy
## Class: A                 1
## Class: B                 1
```

```
## Class: C                1
## Class: D                1
## Class: E                1
```

The fact that the accuracy is 1 is not strange, since the model was built from these data. However, it could indicate overfitting of the model to the train data. To estimate the efficiency of the model better, the prediction of the model was compared to the cross-validation data

```
table(cross$classe,predict(model,cross))
##
##        A    B    C    D    E
##   A 1610    6   19   37    2
##   B   39 1049   37   11    3
##   C   15   19  980    5    7
##   D   17    1   42  899    5
##   E    0   17   13   15 1037
confusionMatrix(cross$classe,predict(model,cross))$overall
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##   9.473237e-01   9.333777e-01   9.413059e-01   9.528930e-01   2.856415e-01
## AccuracyPValue  McnemarPValue
##   0.000000e+00   8.893321e-16
confusionMatrix(cross$classe,predict(model,cross))$byClass
##          Sensitivity Specificity Pos Pred Value Neg Pred Value Precision
## Class: A   0.9577381   0.9845422      0.9611708      0.9831394 0.9611708
## Class: B   0.9606227   0.9812226      0.9209833      0.9909397 0.9209833
## Class: C   0.8974359   0.9904027      0.9551657      0.9769500 0.9551657
## Class: D   0.9296794   0.9867832      0.9325726      0.9861817 0.9325726
## Class: E   0.9838710   0.9906852      0.9584104      0.9964605 0.9584104
##              Recall        F1 Prevalence Detection Rate
## Class: A 0.9577381 0.9594514  0.2854715      0.2734070
## Class: B 0.9606227 0.9403855  0.1855565      0.1782498
## Class: C 0.8974359 0.9254013  0.1855565      0.1665251
## Class: D 0.9296794 0.9311238  0.1643161      0.1527613
## Class: E 0.9838710 0.9709738  0.1790994      0.1762107
##          Detection Prevalence Balanced Accuracy
## Class: A            0.2844520         0.9711402
## Class: B            0.1935429         0.9709227
## Class: C            0.1743415         0.9439193
## Class: D            0.1638063         0.9582313
## Class: E            0.1838573         0.9872781
```

In this case, the *out-of-sample* accuracy is 0.9471538 , which is still very high, suggesting that the model is highly efficient even when confronted with new data sets.

Prediction

```
(answers<-predict(model,test.data[,-160]))
##  [1] B A C A A E D B A A B C B A E E A B B B
## Levels: A B C D E

pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    filename = paste0("problem_id_",i,".txt")
```

```
    write.table(x[i],file=filename,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}

pml_write_files(answers)
```